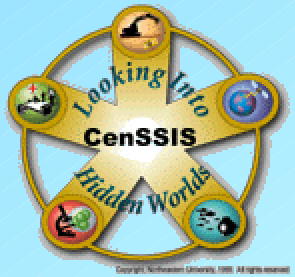
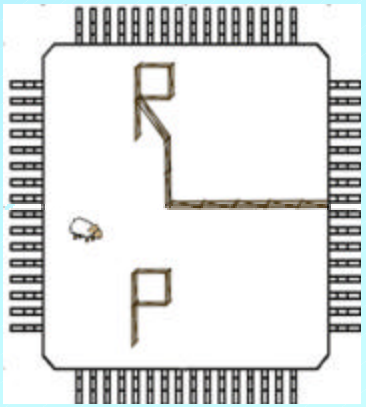
A Library of Parameterized Hardware Modules for Floating Point Arithmetic and Its Use

Prof. Miriam Leeser
Pavle Belanovic
Department of Electrical and Computer Engineering
Northeastern University
Boston MA



Northeastern
UNIVERSITY



Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
21 MAY 2003	**N/A**	**-**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
A Library of Parameterized Hardware Modules for FLoating Point Arithmetic and Its Use	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Department of Electrical and Computer Engineering, Northeastern University, Boston, MA**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**UU**	**27**	

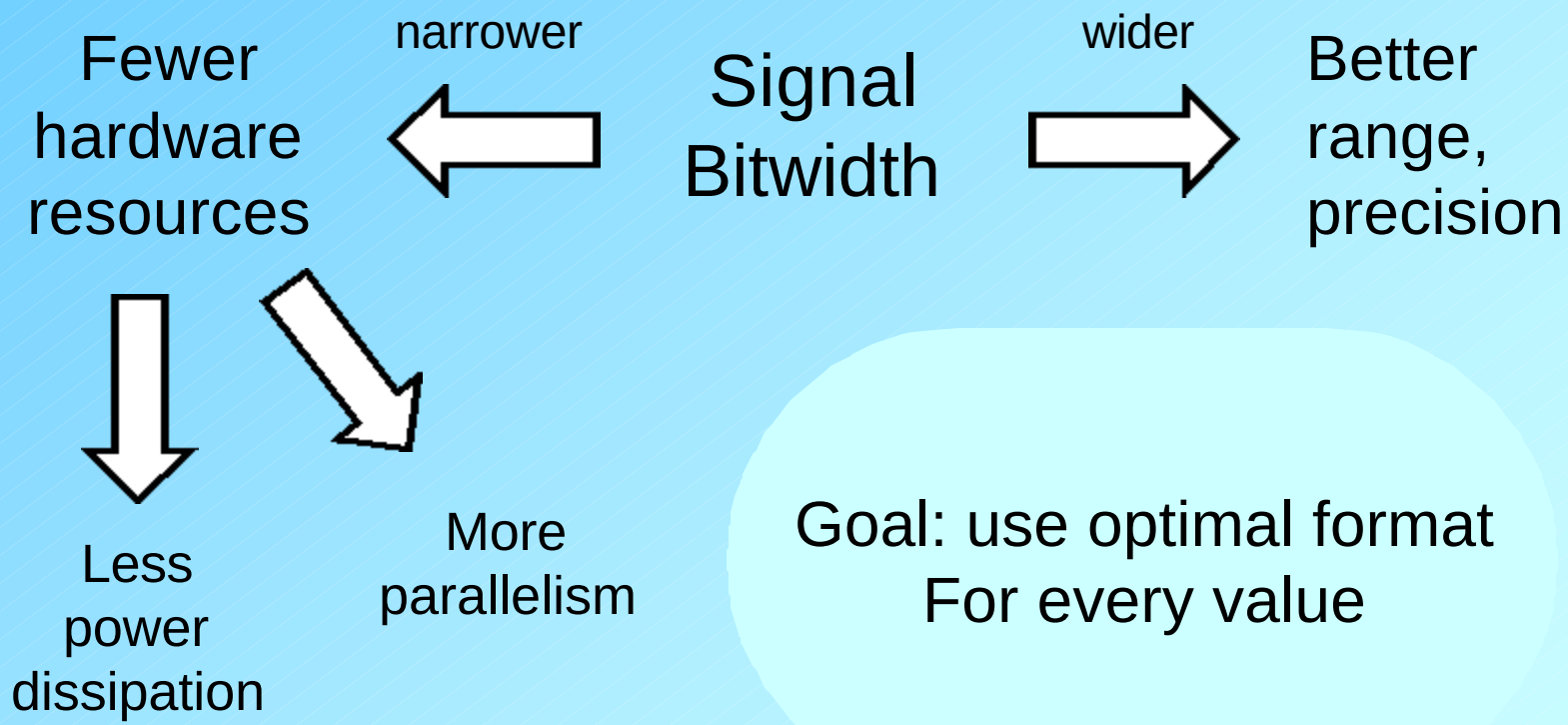
Outline

- Introduction and motivation
- Library of hardware modules for variable precision floating point
- Application: K-means algorithm using variable precision floating point library
- Conclusions

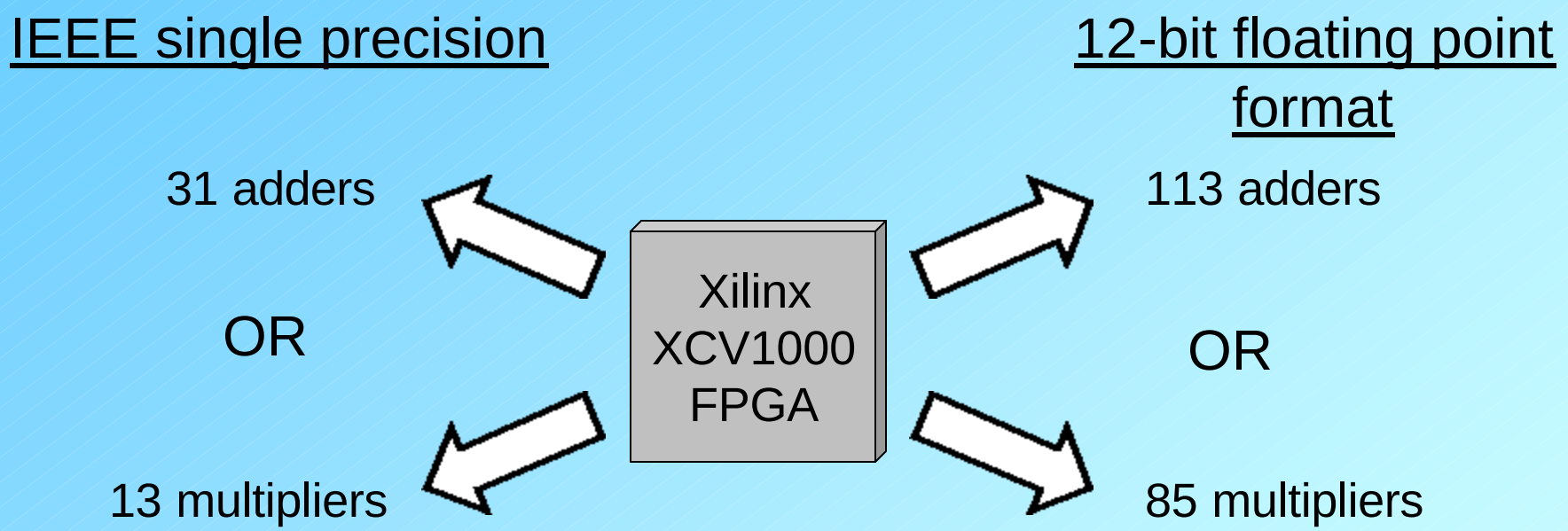
Accelerating Algorithms

- Reconfigurable hardware used to accelerate image and signal processing algorithms
- Exploit parallelism for speedup
- Customize design to fit the particular task
  - signals in fixed or floating-point format
  - area, power vs. range, precision trade-offs

Format Design Trade-offs



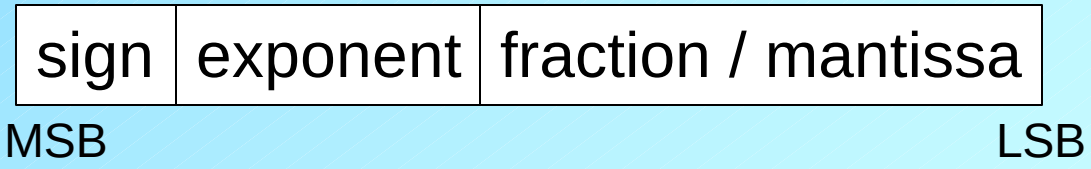
Area Gains Using Reduced Precision



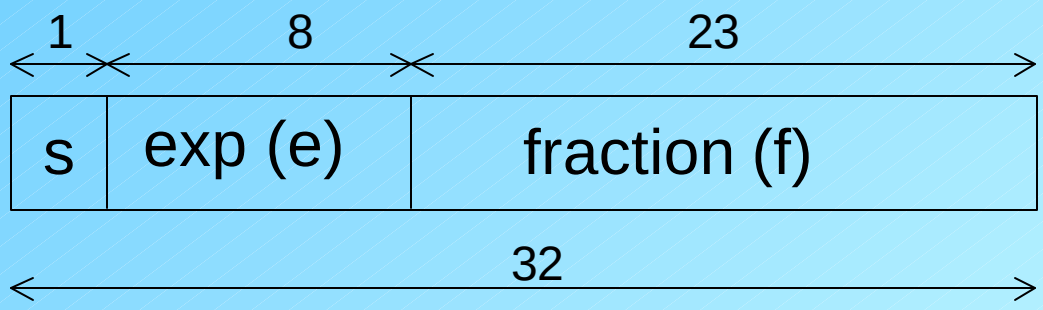
General Floating-Point Format

Field	Symbol	Bitwidth
sign	s	1
exponent	e	*exp_bits*
fraction/mantissa	f	*man_bits*

$$(-1)^{s} * 1.f * 2^{e\text{-}BIAS}$$



IEEE Floating Point Format



$$(-1)^{s} * 1.f * 2^{e-BIAS}$$

- BIAS depends on number of exponent bits
  - » 127 in IEEE single precision format
- Implied 1 in mantissa not stored

Library of Parameterized Modules

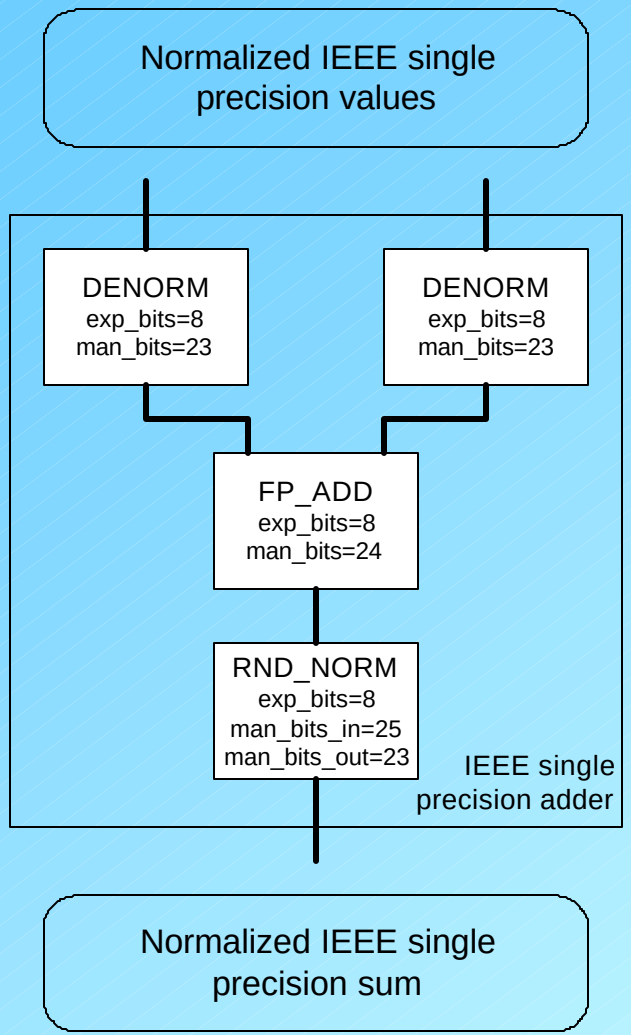
- Total of seven parameterized hardware modules for arbitrary precision floating-point arithmetic

	Module	Latency
format control	denorm	0
	rnd_norm	2
operators	fp_add	4
	fp_sub	4
	fp_mul	3
conversion	fix2float	4 / 5
	float2fix	4 / 5

Highlights

- Completely general floating-point format
- All IEEE formats are a subset
- All previously published non-IEEE formats are a subset
- Abstract normalization from other operations
- Rounding to zero or nearest
- Pipelining signals
- Some error handling

Assembly of Modules



$$2 \times \text{denorm}$$

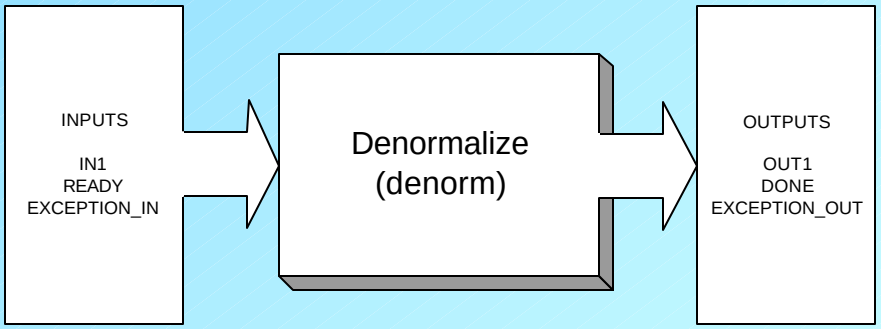
$$+ 1 \times \text{fp\_add}$$

$$+ 1 \times \text{rnd\_norm}$$

$$= 1 \times \text{IEEE single precision adder}$$

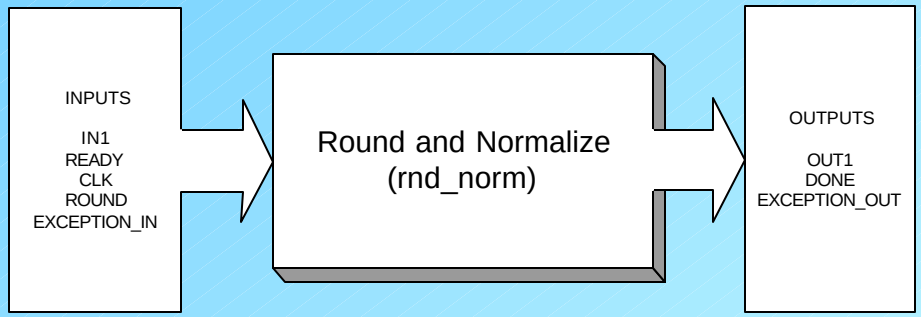
Denormalization

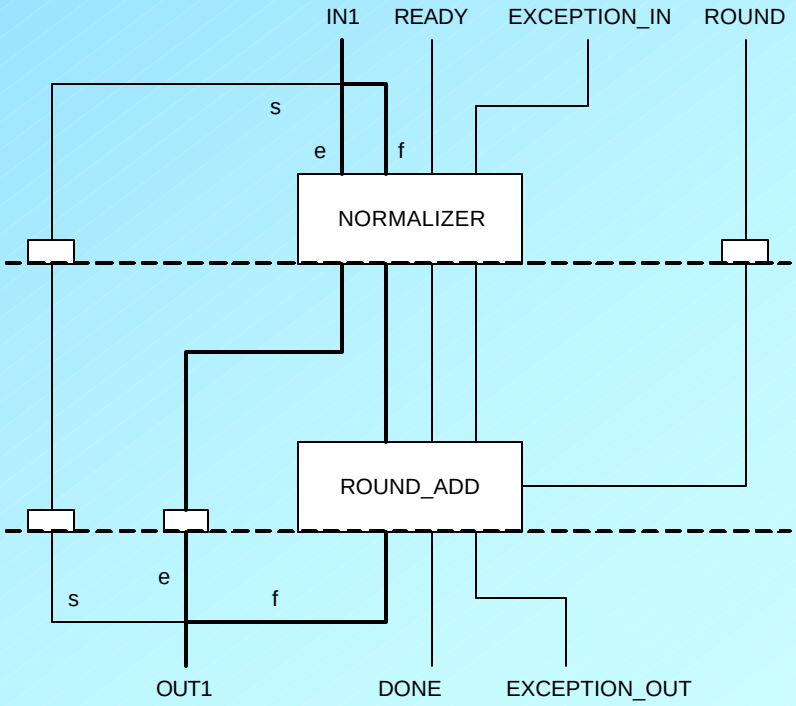
- “Unpack” input number: insert implied digit
- If input is value zero, insert ‘0’
  Otherwise, insert ‘1’
- Output 1 bit wider than input
- Latency = 0



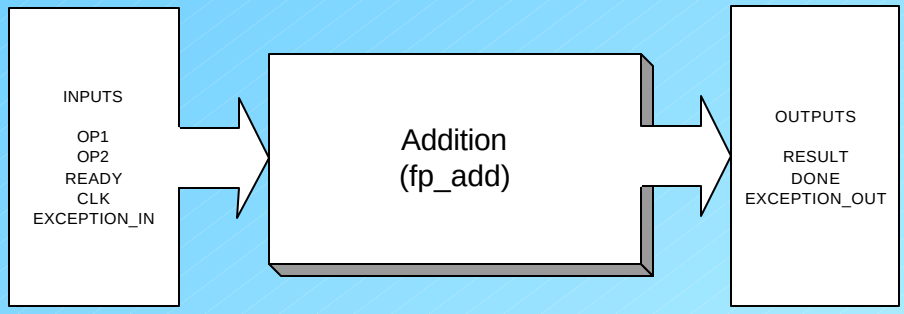
Rounding and Normalizing

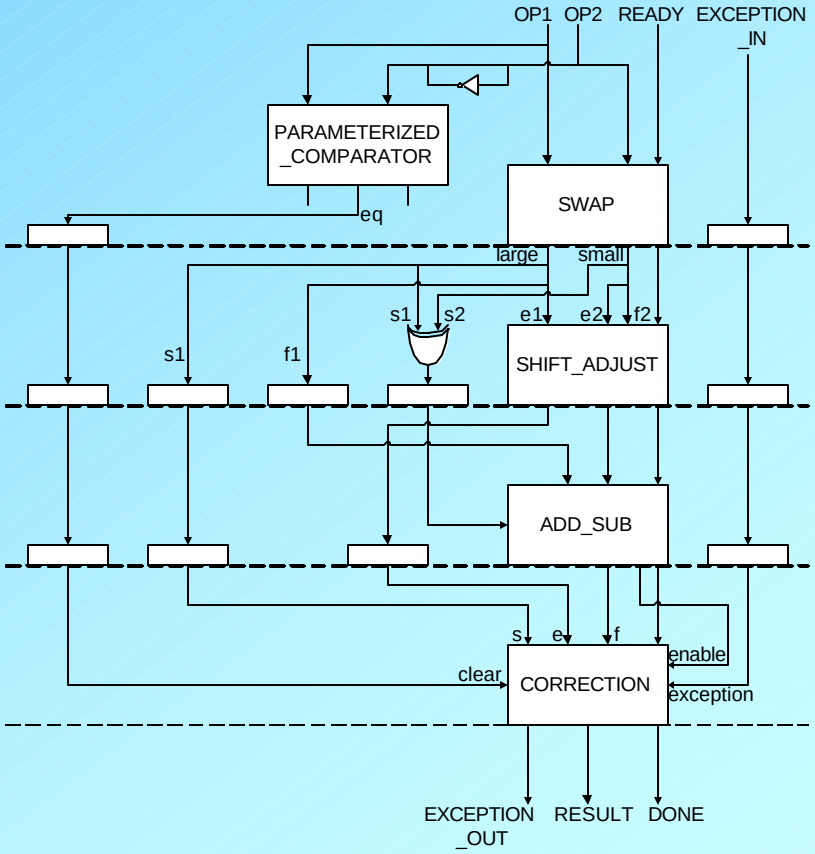
- Returns input to normalized format
- Designed to follow arithmetic operation(s)



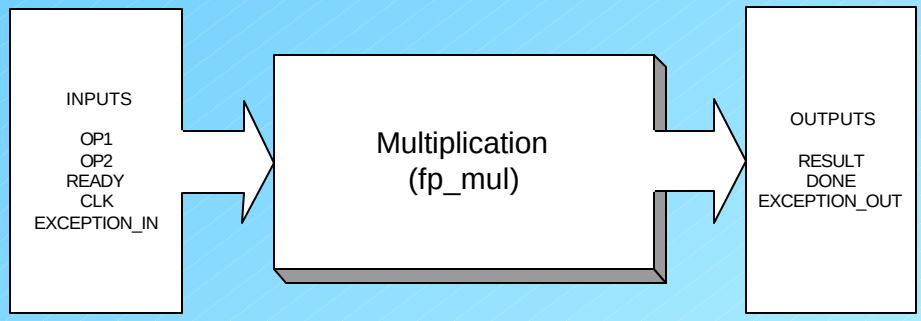


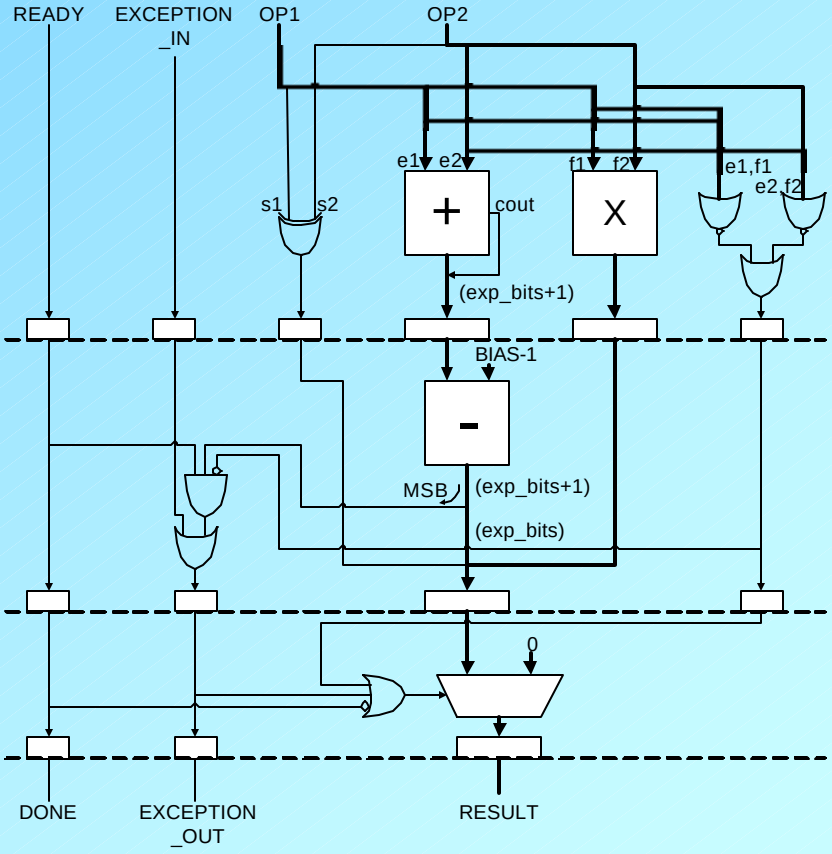
Addition and Subtraction



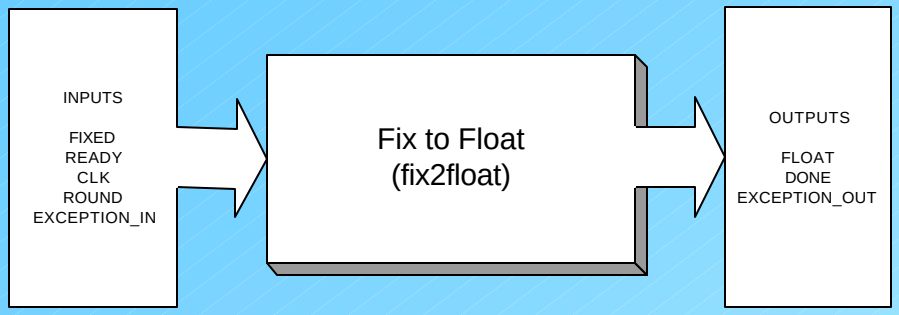


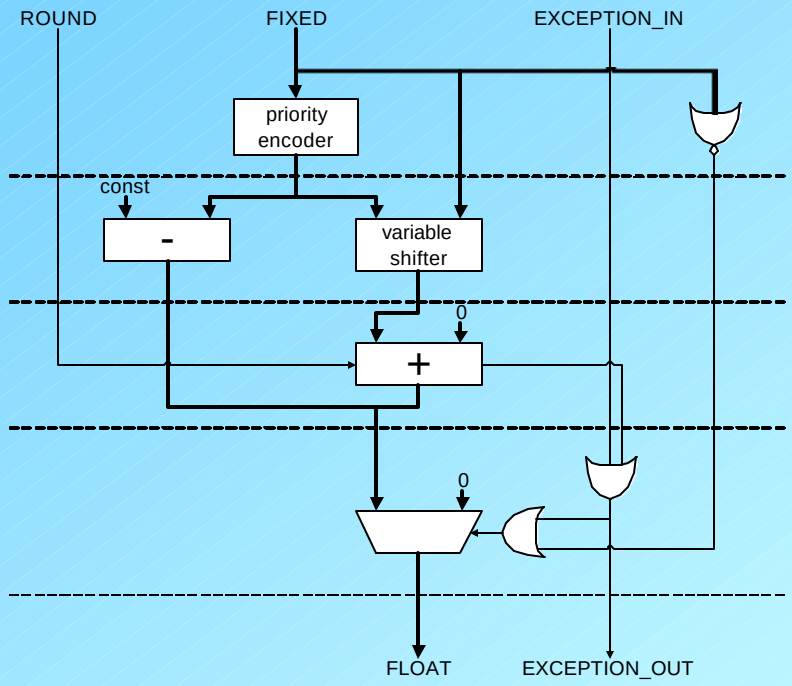
Multiplication

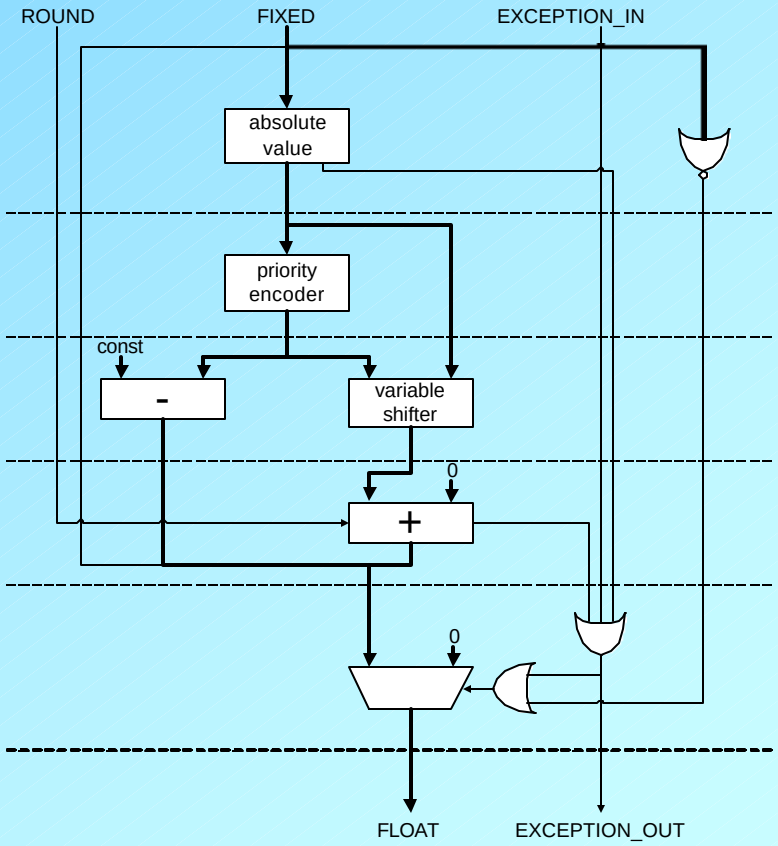




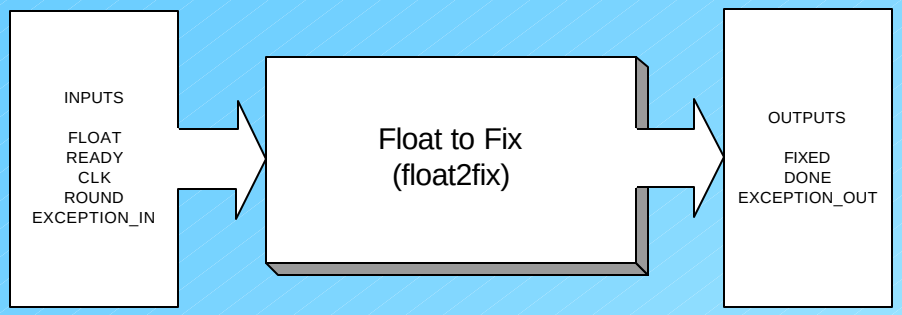
Fixed to Floating-Point

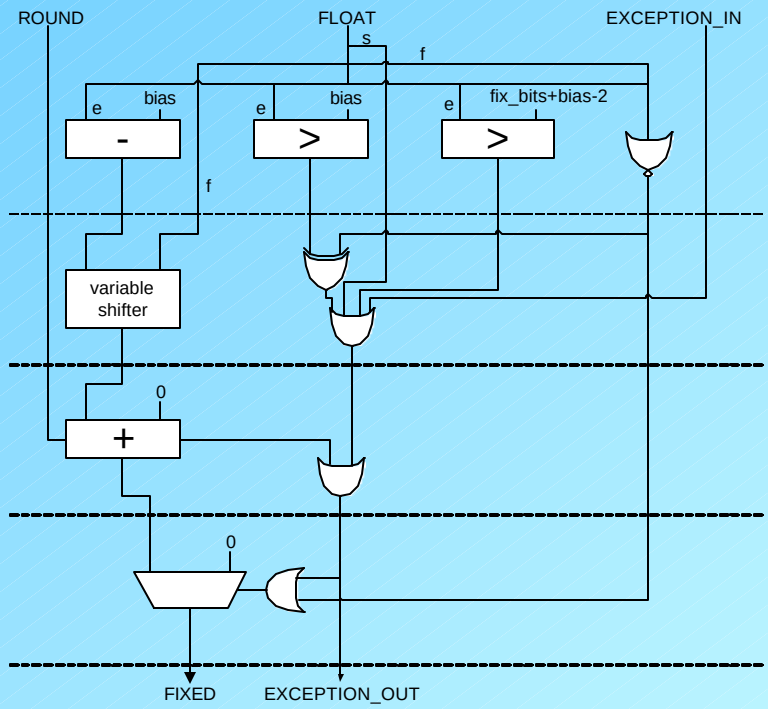


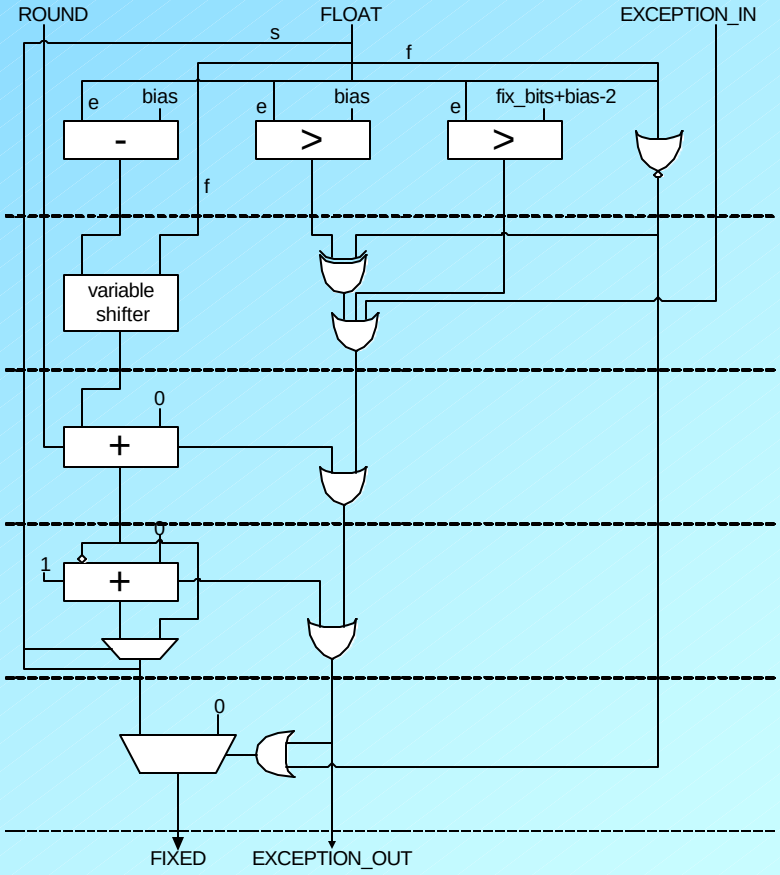




Floating to Fixed-Point

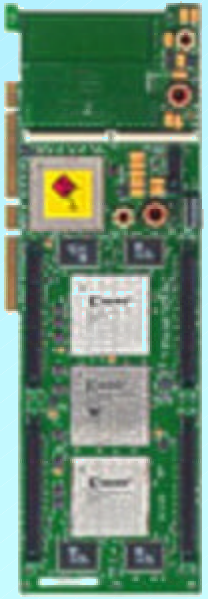




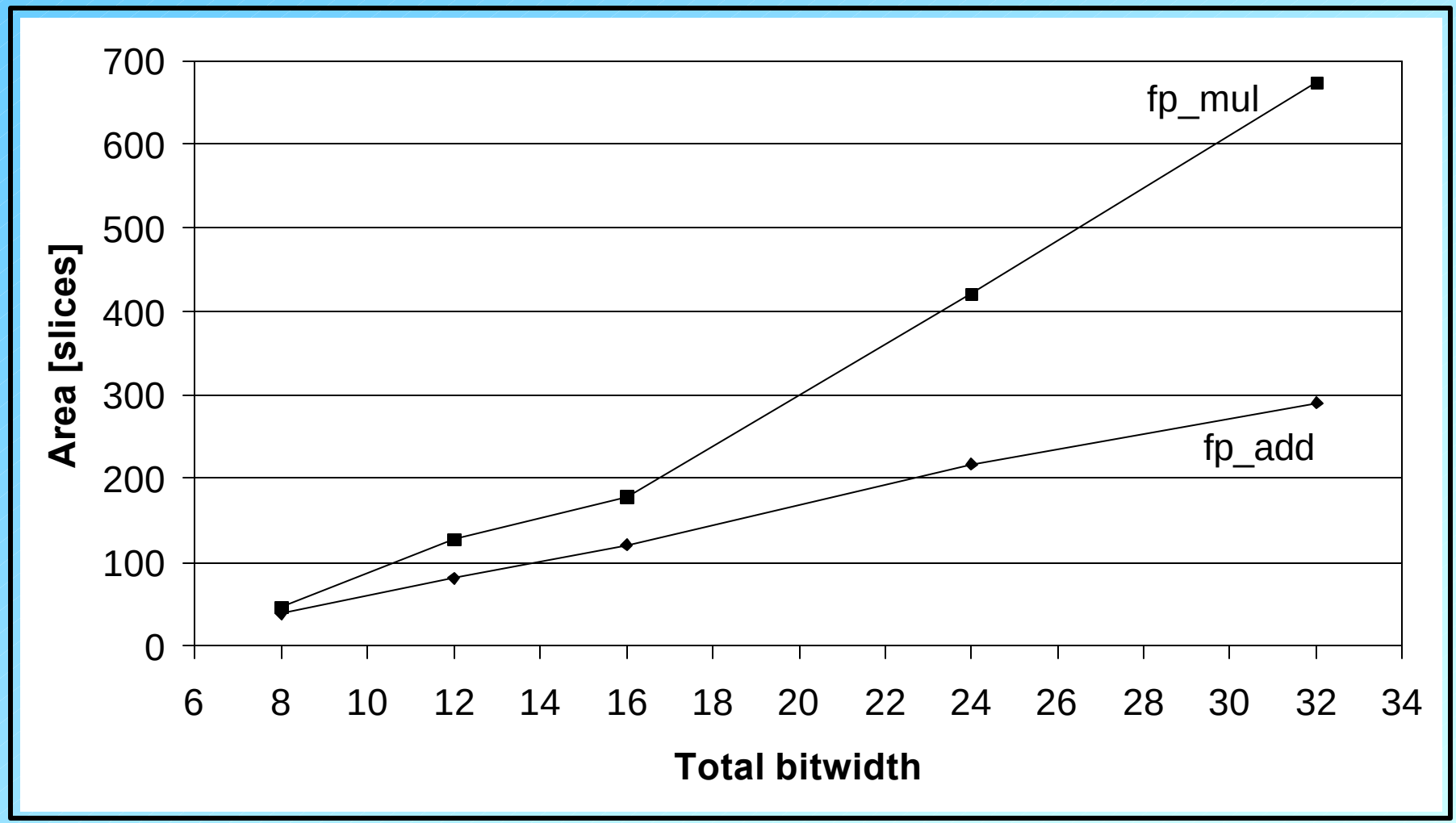


Implementation Experiments

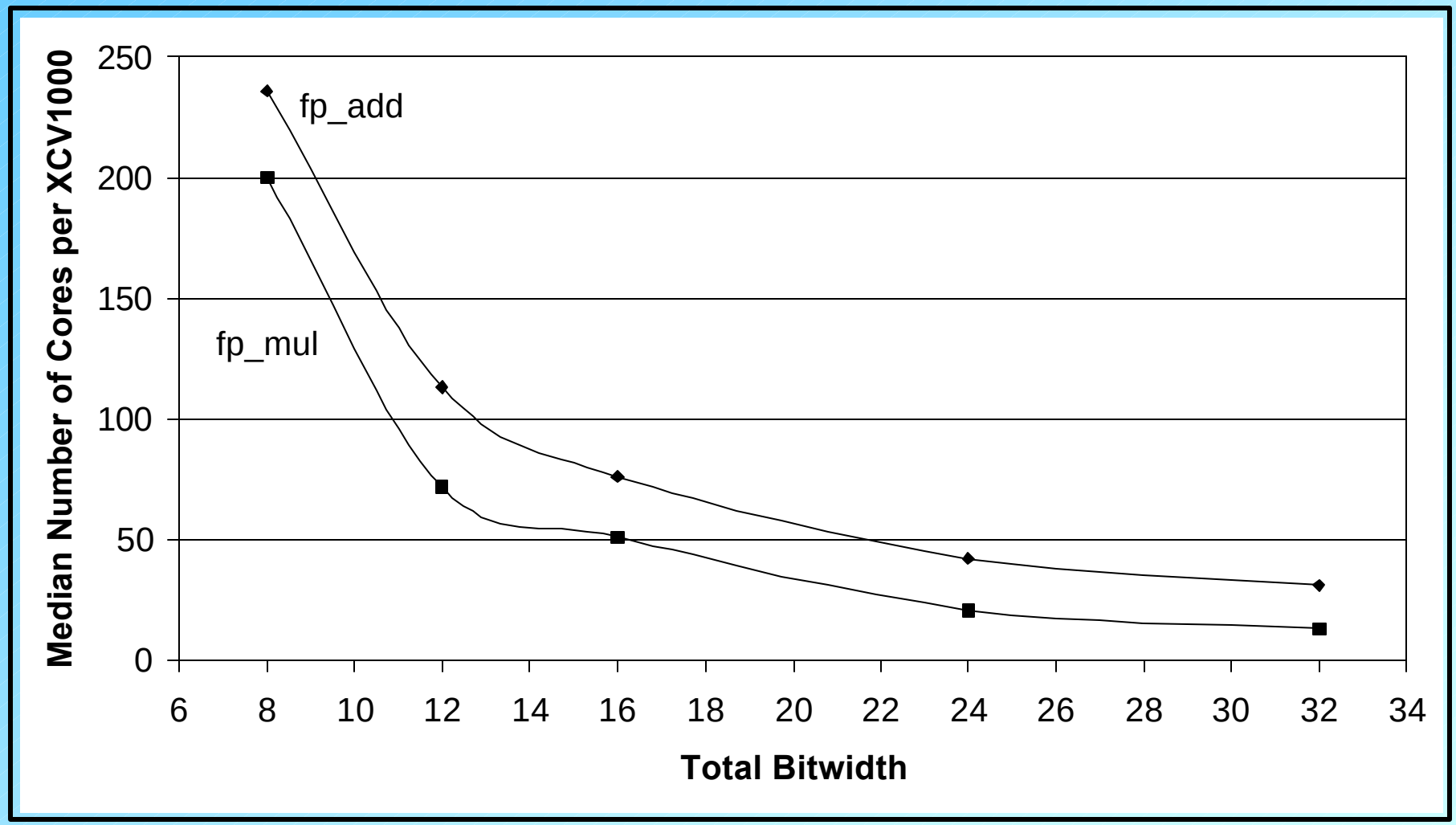
- Designs specified in VHDL
- Mapped to Xilinx Virtex FPGA
- Wildstar reconfigurable computing engine by Annapolis Micro Systems Inc.
  - PCI Interface to host processor
  - 3 Xilinx XCV1000 FPGAs
  - total of 3 million system gates
  - 40 Mbytes of SRAM
  - 1.6 Gbytes/sec I/O bandwidth
  - 6.4 Gbytes/sec memory bandwidth
  - clock rates to 100MHz



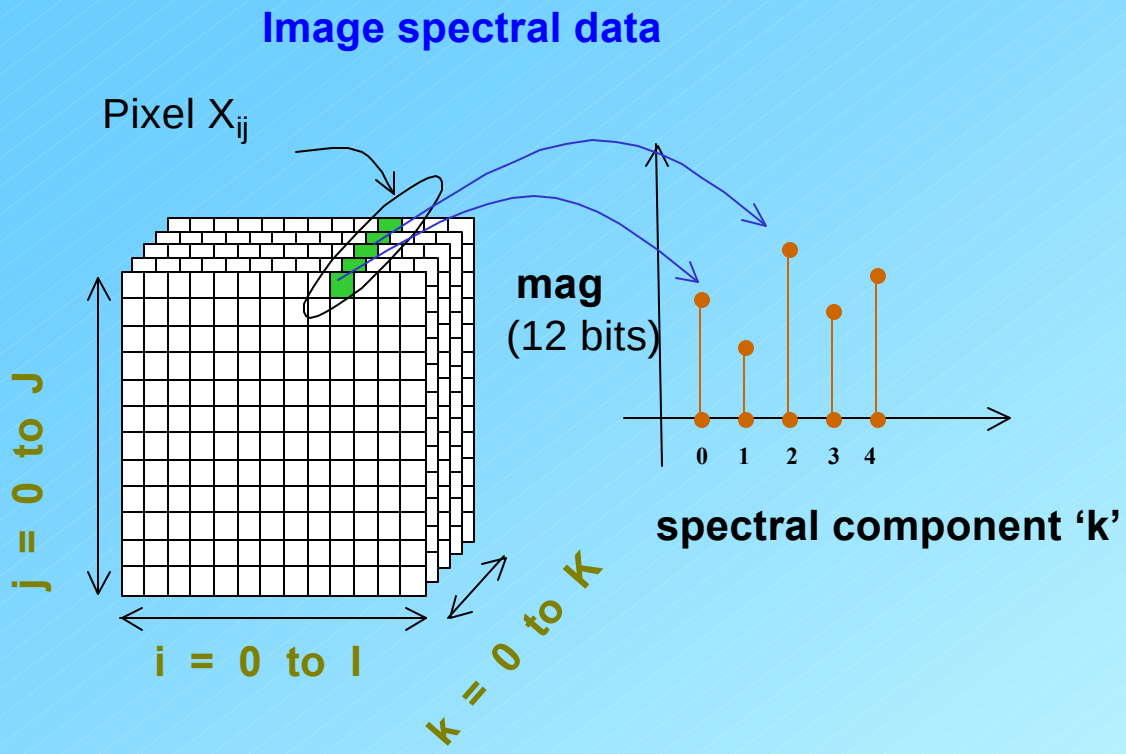
Synthesis Results

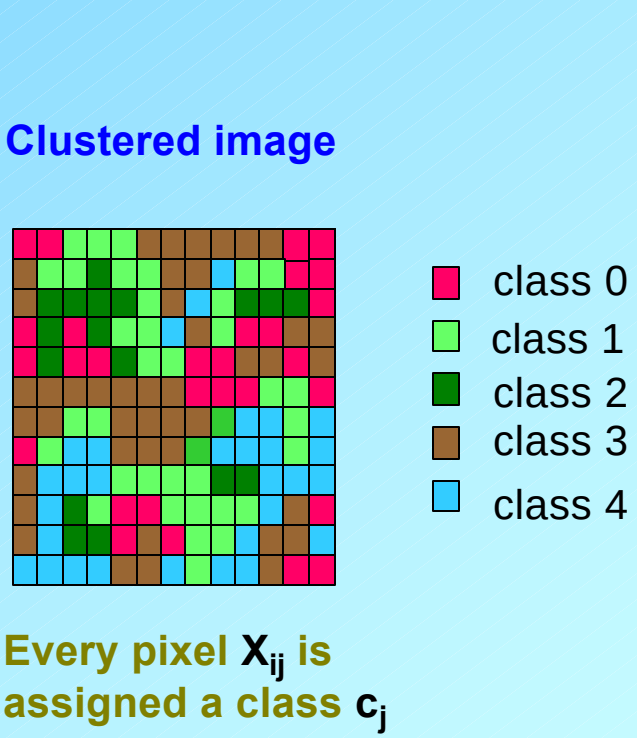


Synthesis Results



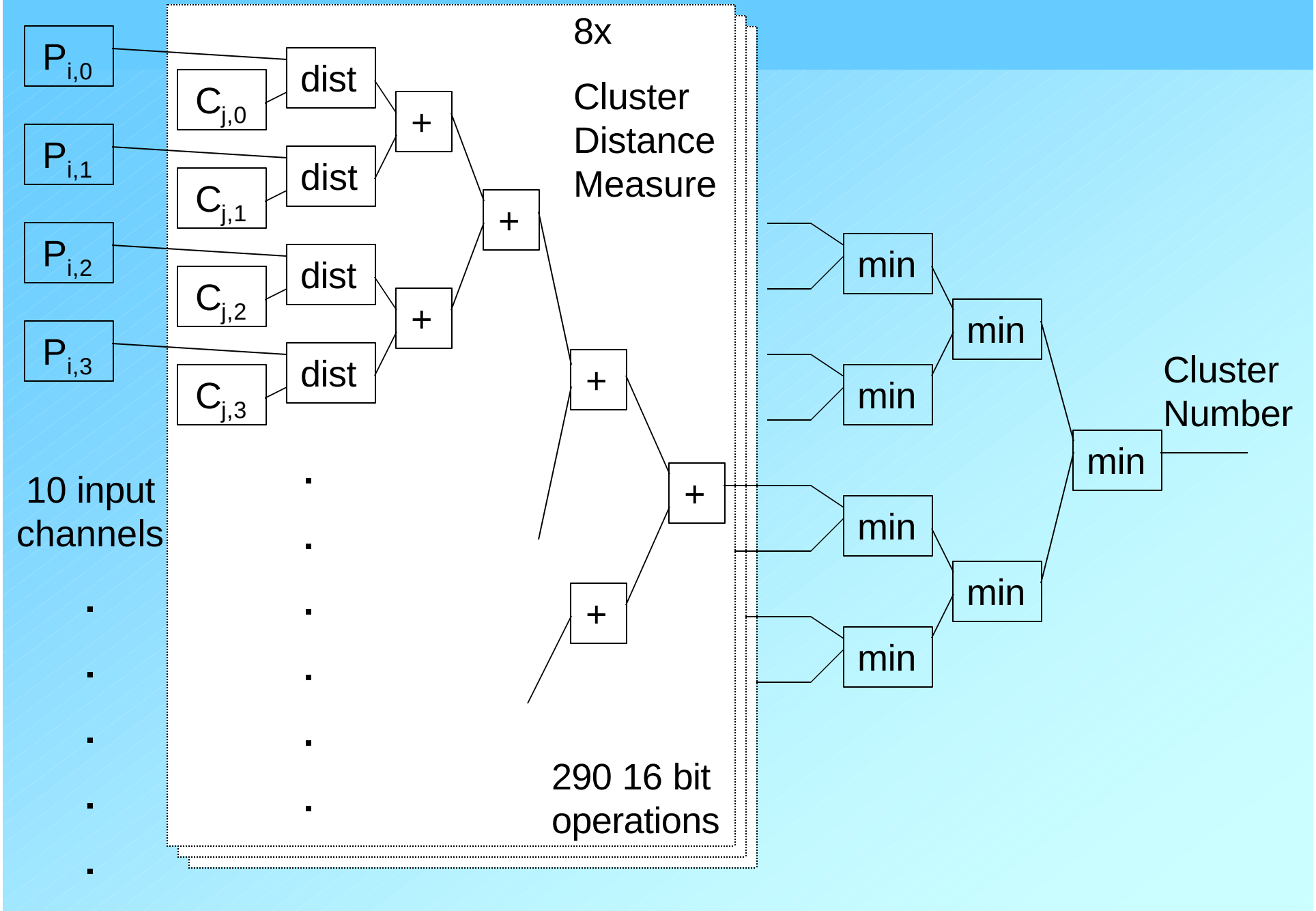
K-means Algorithm





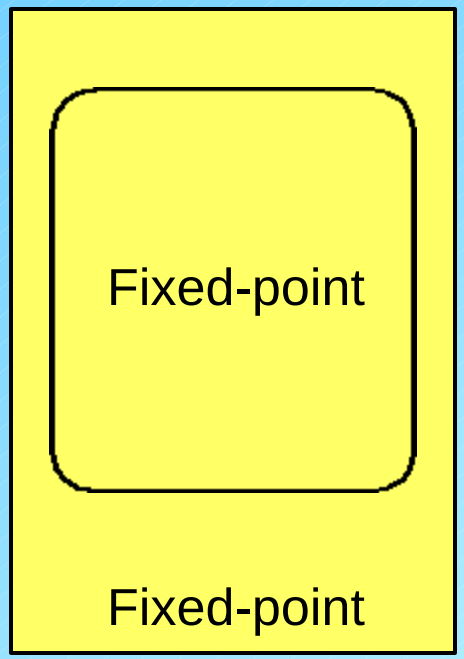
- Each cluster has a center:
  - mean value of pixels in that cluster
- Each pixel is in the cluster whose center it is closest to
  - requires a distance metric
- Algorithm is iterative

Hardware Implementation of k-means Clustering

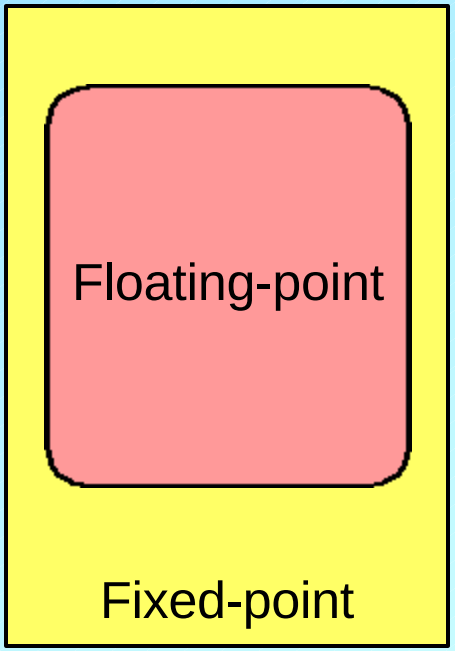


K-means Clustering Algorithm

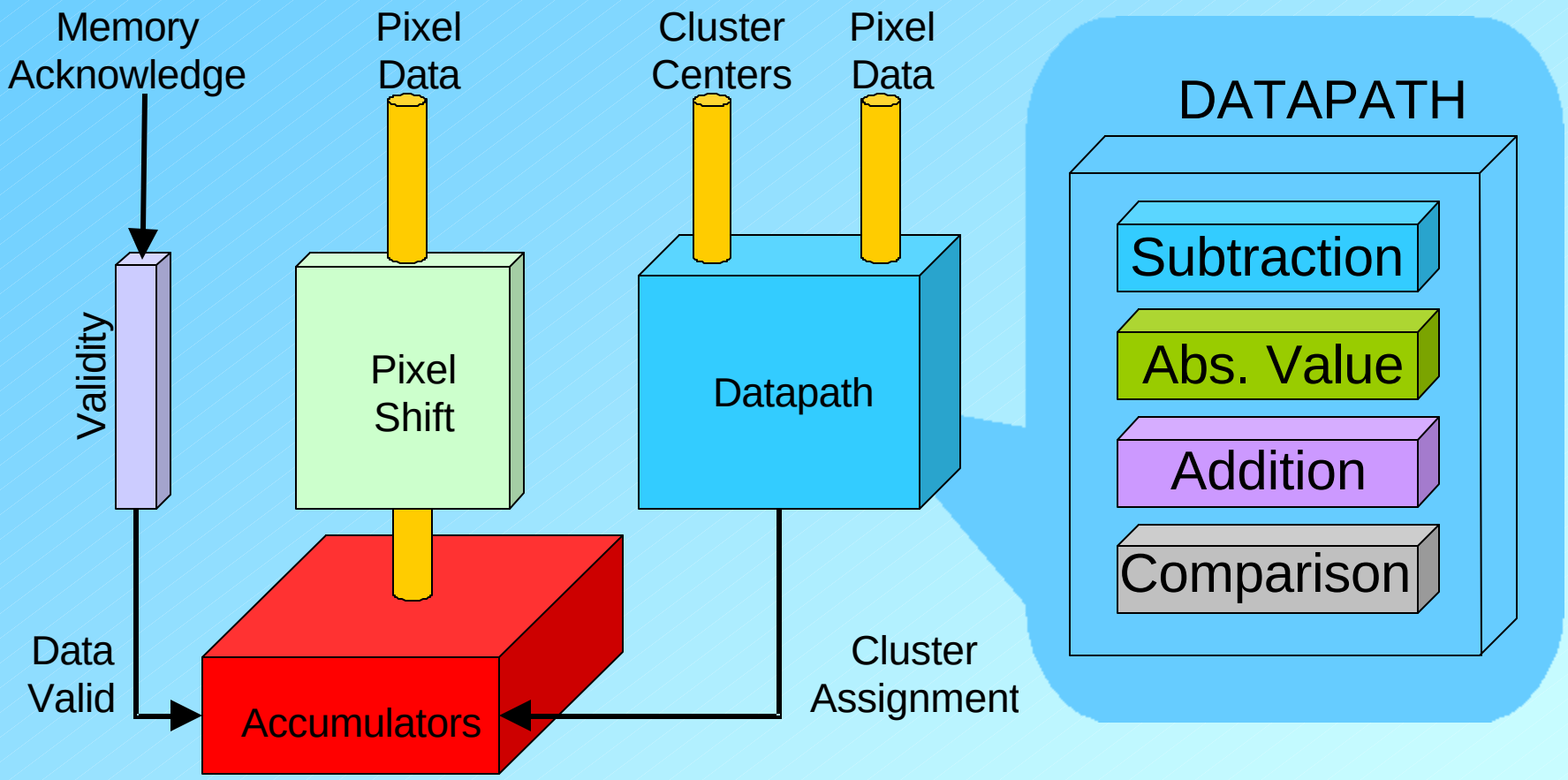
Purely fixed-point



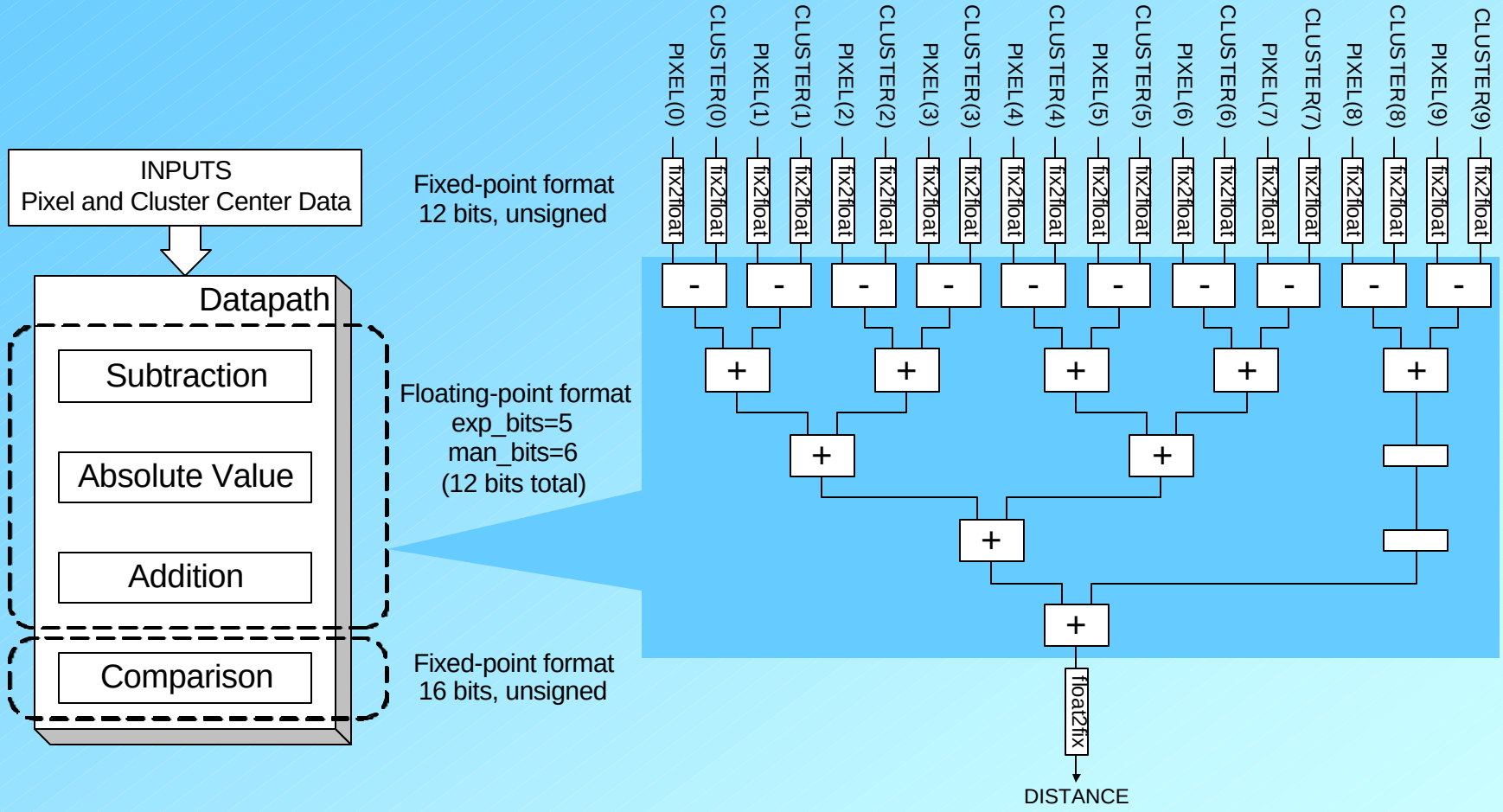
Hybrid fixed and floating-point



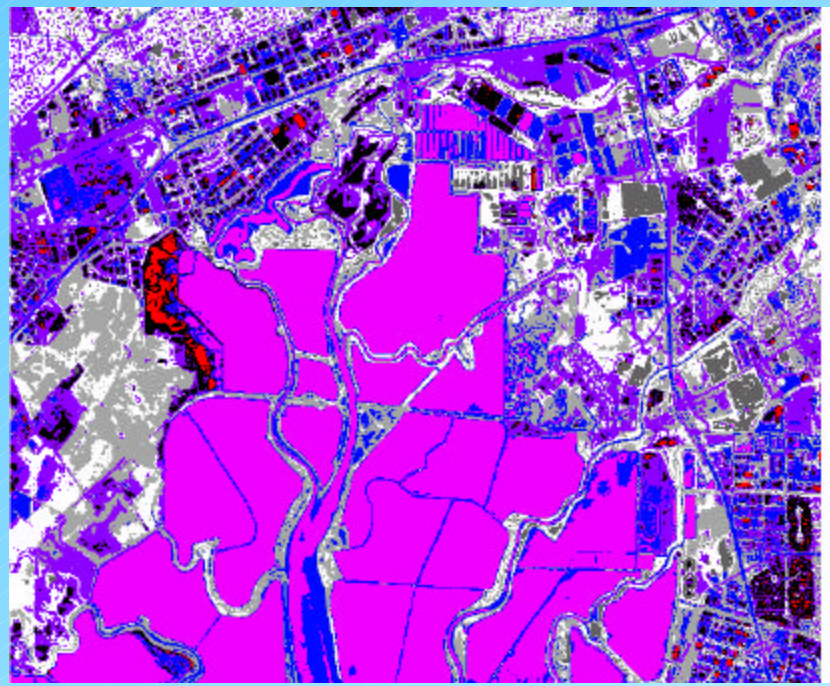
Structure of the K-means Circuit



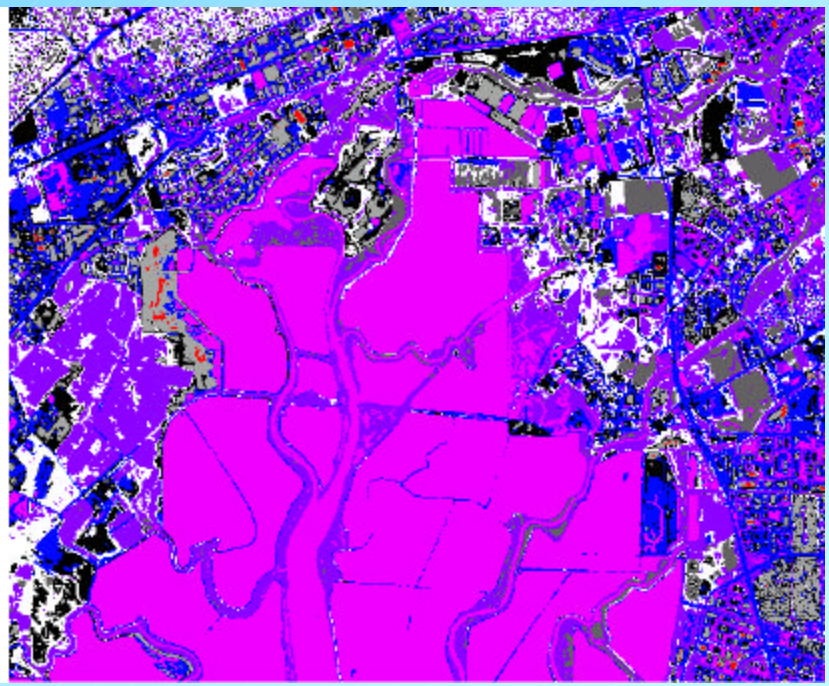
Hybrid Datapath



Results of Processing



Purely fixed-point



Hybrid fixed and floating-point

Synthesis Results

Property	Fixed-point	Hybrid
Area	9420 slices	10883 slices
Percent of FPGA	76%	88%
Minimum period	16ns	20ns
Maximum frequency	64MHz	50MHz
Throughput	1 cycle	8 cycles

Conclusions

- Library of fully parameterized hardware modules for floating-point arithmetic available
- Ability to form arithmetic pipelines in custom floating-point formats demonstrated
- Future work
  - More floating point modules (ACC, MAC, DIV ...)
  - More applications
  - Automation of design process using the library
    - Automatically choose best format for each variable and operation